COPY OF DEPOSIT

TXu 1-990-709

NOTE: deposits submitted electronically bear no identifying marks.

```
class Account < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :lockable, :timeoutable and :omniauthable

  UNDEFINED_CITY = "Undefined city"

  # has_attached_file :account_document, :styles => { :medium => "300x300>", :thumb => "100x10
0>" }, :default_url => "/images/:style/missing.png"
  # validates_attachment_content_type :account_document, :content_type => /\Aimage\/.*\Z/

  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable, :confirmable

  has_many :verifications,      dependent: :destroy
  # has_many :account_documents, dependent: :destroy
  has_one  :wallet,             dependent: :destroy

  belongs_to :country
  belongs_to :state
  belongs_to :city

  validate :check_city

  validates_presence_of :first_name, :last_name, :middle_initials, :email, :phone,
                        :address, :country_id, :city_id

  before_create :set_city

  def to_label
    "#{first_name} #{last_name}"
  end

  def identify_verified?
    verifications.identify.processed.any?
  end

  def financial_verified?
    verifications.financial.processed.any?
  end

  def completely_verified?
    identify_verified? && financial_verified?
  end

  def admin?
    account_type == "admin" ? true : false
  end

  def verify( verification )
    verifications << verification
  end

  def banned?
    !active?
  end

  def to_hash
    { first_name: first_name, last_name: last_name, middle_initials: middle_initials, email: e
mail,
      phone_number: phone, address: address, country_id: country_id, state_id: state_id, city_
id: city_id,
      created_at: created_at.to_i, updated_at: updated_at.to_i
    }
  end
```

```
  def active_for_authentication?
    super && !self.banned?
  end

  protected

  def set_city
    city = City.find_by_name(city_id)
    self.city_id = city.id
  end

  def check_city
    self.errors.add(:base, UNDEFINED_CITY) if city_id && City.find_by_name(city_id).nil?
  end

end
```

```
class Api::Main

  API_EXTERNAL_SECRET_KEY = "273tegdewjfbcbdsjdkadkjasndjkaddsad"

  def initialize(params, request)
    @params = params
    @request = request
  end

  def create_response
    write_params_in_file

    begin
      encrypted_data =  @params["data"]
      decrypted_data = decrypt(encrypted_data)

      data_hash = eval( decrypted_data)

      cmd = data_hash[:cmd]
      response = send( cmd, data_hash.except(:cmd) )

      { result: "success", data: encrypt( response.to_s ) }

    rescue Exception => e
      { result: "error", message: e.message }
    end

  end

  protected

  def decrypt( data )
    cipher = Gibberish::AES.new( API_EXTERNAL_SECRET_KEY )
    cipher.dec(data)
  end

  def encrypt( data )
    cipher = Gibberish::AES.new( API_EXTERNAL_SECRET_KEY )
    cipher.enc( data )
  end

  def bgcw_get_updated_accounts( params )
    time_to = Time.at( params[:time_to].to_i )
    time_from = Time.at( params[:time_from].to_i )
    Wallet.get_accounts(time_from, time_to)
  end


  # def bgcw_get_countries( params )
  #   time_to = Time.at( params[:time_to].to_i )
  #   time_from = Time.at( params[:time_from].to_i )
  #   Country.where(updated_at: time_from..time_to ).to_a.map(&:serializable_hash)
  # end

  # def bgcw_get_states( params )
  #   time_to = Time.at( params[:time_to].to_i )
  #   time_from = Time.at( params[:time_from].to_i )
  #   State.where(updated_at: time_from..time_to ).to_a.map(&:serializable_hash)
  # end

  # def bgcw_get_cities( params )
  #   time_to = Time.at( params[:time_to].to_i )
  #   time_from = Time.at( params[:time_from].to_i )
  #   City.where(updated_at: time_from..time_to ).to_a.map(&:serializable_hash)
  # end
```

```
  def write_params_in_file
    fname = "test.txt"
    file = File.open(fname, "w")
    file.puts "#{@params}\n"
    file.close
  end

end
```

```
class Verification < ActiveRecord::Base
  belongs_to :account

  scope :processed, ->{ where( status: "processed" ) }
  scope :financial, ->{ where( verification_type: "directid" )}
  scope :identify,  ->{ where( verification_type: "idchecker" )}

  def to_label
    "#{verification_type.humanize} verification for #{account.email}"
  end

end
```

```
class AccountDocument < ActiveRecord::Base
  # belongs_to :account

end
```

```
class Wallet < ActiveRecord::Base
  belongs_to :account

  validates_presence_of :username
  validates_presence_of :password


  def to_label
    "Walet for #{account.email}"
  end

  def self.generate_username
    ('a'..'z').sort_by {rand}[0,2].join.upcase + (1..9).sort_by {rand}[0,5].join
  end

  def self.generate_password
    SecureRandom.hex(5)
  end

  def self.create_wallet( account )
    create( account: account, username: generate_username, password: generate_password )
  end

  def self.get_accounts(time_to, time_from)
    includes(:account).where(updated_at: time_to..time_from  ).all.map do | wallet |
      { login: wallet.username, password: wallet.password }.merge(wallet.account.to_hash)
    end
  end

end
```

```
class Api::MainController < ApplicationController

  skip_before_filter :verify_authenticity_token

  def call
    @response = Api::Main.new( params, request ).create_response
    respond_to do |format|
      format.json { render json: @response}
    end
  end

  def server_x_callback
    response = ServerXApi::Main.process_callback( params )
    render text: response
  end

  private

  def write_params_in_file
    fname = "test.txt"
    file = File.open(fname, "w")
    file.puts "#{params}\n"
    file.close
  end

end
```

```
class AccountsController < ApplicationController

  before_action :set_current_account, only: [:profile, :wallet, :update_wallet]
  before_action :set_wallet_for_current_account, only: [:wallet]
  # before_action :check_all_verifications_steps, only: [:wallet, :update_wallet]

  def profile
  end

  def change_password
  end

  def wallet # GET /control/wallet
  end

  def update_wallet # POST /wallet
    @wallet = Wallet.new( wallet_params )
    if @wallet.valid?
      @account.wallet.update_attributes( wallet_params )
      redirect_to( :control, notice: "You have successfully updated your wallet." )
    else
      @errors = @wallet.errors.full_messages
      set_wallet_for_current_account
      render :wallet
    end
  end

  private

  # def check_all_verifications_steps
  #   redirect_to(:control, alert: "Sorry, service is temporarily unavailable") unless @accoun
t.completely_verified?
  # end

  def wallet_params
    params.require(:wallet).permit( :username, :password )
  end

  def set_wallet_for_current_account
    wallet = @account.wallet
    @wallet = wallet.present? ? wallet : Wallet.create_wallet( @account )
  end

  def set_current_account
    @account = current_account
  end

end
```

```
class Admin::AccountsController < Admin::AdminController
  active_scaffold :account do |config|

    config.actions = [:list, :create, :delete, :update, :show,:search]

    config.columns = [:id, :first_name, :last_name, :middle_initials, :email, :phone, :address
, :active, :created_at]

    config.create.columns = [:id, :first_name, :last_name, :middle_initials, :phone, :address,
 :active, :email, :password]

    # config.list.columns = [:id, :first_name, :last_name, :mid_initials, :email, :phone, :add
ress, :active, :created_at]

    # config.show.columns = [:id, :first_name, :last_name, :mid_initials, :email, :phone, :add
ress, :active, :created_at]


    # config.update.columns = [:login, :email, :first_name, :last_name, :active, :phone_number
]

    # config.show.columns = [:login, :encryped_password, :email, :first_name, :last_name, :act
ive, :phone_number, :created_at ]

  end
end
```

```
class Admin::AdminController < ApplicationController
  before_filter :authenticate_admin_account
  skip_before_filter :check_all_steps_verirification

  protected

  def authenticate_admin_account
    redirect_to root_path unless current_account && current_account.admin?
  end

end
```

```
class Admin::VerificationsController < Admin::AdminController
    active_scaffold :verification do |config|
      config.actions = [:list, :create, :delete, :update, :show,:search]

      config.list.columns = [:id, :account, :verification_type, :status, :cancel_reason, :crea
ted_at]

      config.create.columns = [:account, :verification_type, :status, :cancel_reason ]
    end
end
```

```
class Admin::WalletsController < Admin::AdminController
    active_scaffold :wallet do |config|
      config.actions = [:list, :create, :delete, :update, :show,:search]
    end
end
```

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  before_filter :check_all_steps_verirification, unless: :devise_controller?
  before_filter :configure_permitted_parameters, if: :devise_controller?
  before_filter :set_gflash

  VERIFICATIONS_MSG = "You should first be verified."


  private

  def set_gflash
    gflash :success => flash[:notice] if flash[:notice]
    gflash :error => flash[:alert] if flash[:alert]
  end

  def configure_permitted_parameters
    devise_parameter_sanitizer.for(:sign_up) << [:first_name, :last_name, :middle_initials, :a
ddress, :phone, :active, :city_id, :state_id, :country_id, :zip]
    # devise_parameter_sanitizer.for(:account_update) << :first_name
  end

  def check_all_steps_verirification
    if current_account
      redirect_to :verification , alert: VERIFICATIONS_MSG unless current_account.completely_v
erified?
    end
  end

end
```

```
class LocationsController < ApplicationController

  def get_states # POST
    country = Country.find_by_code(params[:country_id])
    @states = country.states.order("name")
    render partial: "get_states", locals: { states: @states }
  end

  def get_cities # POST
    country = Country.find_by_code(params[:country_id])
    @cities = country.cities
    city = params[:account] && params[:account][:city_id] ? params[:account][:city_id] : ""
    render partial: "get_cities", locals: { cities: @cities, city: city }
  end

  def search_city
    country = Country.find_by_code(params[:country_id])

    cities = country.cities.where("Name LIKE ?", "#{params[:term]}%" ).order("Name")

    result = cities.map{ | city | { value: city.name } }

    render json: result
  end
end
```

```
class WelcomeController < ApplicationController
  def index
  end
end
```

```
class SessionsController < Devise::SessionsController
end
```

```
require 'uri'
require 'net/http'

class VerificationsController < ApplicationController

  SUCCESS_RESPONSE_STATUS = "success"

  # ID_CHECKER_URL = "https://demo.idchecker.com/IDchecker.WebAPI.StrongID/Api/v2/StrongID/"
  ID_CHECKER_URL = "https://accplatform.idchecker.nl/idchecker.webapi.strongid/api/v2/strongid
/"

  TEST_EMAIL = "marcus@atencoin.com"
  TEST_API_KEY = "jOd+52JfzOPd4sUmUCPrdYs4dQZvL2ywGRpwXpZlnLM="


  before_action :set_user, only: [:miicard_verification]

  before_action :authenticate_account!

  skip_before_filter :check_all_steps_verirification

  def miicard_verification
    opts = { id: @account.id.to_s, first_name: @account.first_name, last_name: @account.last_n
ame }
    response = ServerXApi::Main.send_data( opts )
    if response["result"] == SUCCESS_RESPONSE_STATUS
      @token = response["data"]["token"]
      render partial: "miicard", locals: { token: @token }
    else
      redirect_to :verification, alert: "Service is temporarily unavailable."
    end
  end

  def check_verifications
  end

  def idchecker_verification
    data_image =  Base64.encode64(File.read(params[:account][:account_document].tempfile)).gsu
b("\n", '')


    opts = { client_reference: current_account.id.to_s, data_image: data_image, callback_uri:
verifications_identify_url }
    authentication_opts = { username: params[:username], api_key: params[:api_key] }
    # authentication_opts = { username: TEST_EMAIL, api_key: TEST_API_KEY }

    response = IdChecker::Main.new.send_request( ID_CHECKER_URL , opts.merge(authentication_op
ts) )
    # render text: response.inspect
    redirect_to :verification, alert: "Service is temporarily unavailable."
  end

  def identify
    # @url = JUMIO_REQUEST_URL
    @url = verifications_identify_path
  end

  def idchecker_callback
    render text: params
  end

  private

  def set_user
    @account = current_account
```

```
module EncryptFactory

  def encrypt_data( data, key, algorithm )
    cipher = OpenSSL::Cipher.new( algorithm )
    cipher.encrypt
    cipher.key = key

    code = cipher.update(data) + cipher.final

    base64_encode( code )
  end

  def decrypt_data( encrypted_data, key, algorithm )
    decode = base64_decode( encrypted_data )

    decipher = OpenSSL::Cipher.new( algorithm )
    decipher.decrypt
    decipher.key = key

    decipher.update(decode) + decipher.final
  end

  def make_signature( data, token)
    Digest::MD5.hexdigest( data.keys.sort.map { |key| data[key] }.inject(:+) + token )
  end

  private

  def base64_encode(code)
    Base64.encode64(code)
  end

  def base64_decode(encrypted_data)
    Base64.decode64(encrypted_data)
  end


end
```

```
require 'net/http'
require 'uri'

class Sender

  attr_reader :uri

  def initialize( url )
    @uri = URI.parse(url)
  end

  def post_data( data )
    req = Net::HTTP::Post.new(@uri)
    req.set_form_data( data )

    response = Net::HTTP.start(@uri.host, @uri.port, use_ssl: https?) do |http|
      http.request(req)
    end

    response.body

  end

  def get_data
    response = Net::HTTP.start(@uri.host, @uri.port, use_ssl: https?) do |http|
      request = Net::HTTP::Get.new(@uri)
      http.request(request)
    end

    response.body
  end

  private

  def https?
    @uri.scheme == "https"
  end

end
```

```
Rails.application.routes.draw do
  get 'locations/get_states'

  devise_for :accounts, :controllers => {:sessions => "sessions"}
  # The priority is based upon order of creation: first created -> highest priority.
  # See how all your routes lay out with "rake routes".

  # You can have the root of your site routed with "root"
  root 'welcome#index'

  # get "/control" => "welcome#index"
  get "/control/profile" => "accounts#profile"
  get "/control/change-password" => "accounts#change_password"
  get "/control/wallet" => "accounts#wallet"


  post "/wallet" => "accounts#update_wallet"

  namespace :admin do
    resources :accounts, :wallets, :verifications do
      as_routes
    end
  end

  namespace :api, :defaults => {:format => :json} do
    post "/" => "main#call"
    post "/server-x" => "main#server_x_callback"
  end

  post "/locations/get_states" => "locations#get_states"
  post "/locations/get_cities" => "locations#get_cities"

  resources :locations, only: [] do
    get :search_city, :on => :collection
  end

  post "/verifications/financial" => "verifications#miicard_verification"

  get "/verification" => "verifications#check_verifications"

  get "/verification/identify" => "verifications#identify"

  post "/verification/identify-callback" => "verification#idchecker_callback"

  post "/verifications/identify" => "verifications#idchecker_verification"
end
```

```
GEM
  remote: https://rubygems.org/
  specs:
    actionmailer (4.2.0)
      actionpack (= 4.2.0)
      actionview (= 4.2.0)
      activejob (= 4.2.0)
      mail (~> 2.5, >= 2.5.4)
      rails-dom-testing (~> 1.0, >= 1.0.5)
    actionpack (4.2.0)
      actionview (= 4.2.0)
      activesupport (= 4.2.0)
      rack (~> 1.6.0)
      rack-test (~> 0.6.2)
      rails-dom-testing (~> 1.0, >= 1.0.5)
      rails-html-sanitizer (~> 1.0, >= 1.0.1)
    actionview (4.2.0)
      activesupport (= 4.2.0)
      builder (~> 3.1)
      erubis (~> 2.7.0)
      rails-dom-testing (~> 1.0, >= 1.0.5)
      rails-html-sanitizer (~> 1.0, >= 1.0.1)
    active_scaffold (3.4.17)
      rails (>= 3.2.18, < 5)
    activejob (4.2.0)
      activesupport (= 4.2.0)
      globalid (>= 0.3.0)
    activemodel (4.2.0)
      activesupport (= 4.2.0)
      builder (~> 3.1)
    activerecord (4.2.0)
      activemodel (= 4.2.0)
      activesupport (= 4.2.0)
      arel (~> 6.0)
    activesupport (4.2.0)
      i18n (~> 0.7)
      json (~> 1.7, >= 1.7.7)
      minitest (~> 5.1)
      thread_safe (~> 0.3, >= 0.3.4)
      tzinfo (~> 1.1)
    aescrypt (1.0.0)
    arel (6.0.0)
    autoprefixer-rails (5.1.6)
      execjs
      json
    bcrypt (3.1.10)
    bootstrap-sass (3.3.3)
      autoprefixer-rails (>= 5.0.0.1)
      sass (>= 3.2.19)
    builder (3.2.2)
    climate_control (0.0.3)
      activesupport (>= 3.0)
    cocaine (0.5.5)
      climate_control (>= 0.0.3, < 1.0)
    coffee-rails (4.0.1)
      coffee-script (>= 2.2.0)
      railties (>= 4.0.0, < 5.0)
    coffee-script (2.3.0)
      coffee-script-source
      execjs
    coffee-script-source (1.9.0)
    crypt-api (0.2.5)
    devise (3.4.1)
      bcrypt (~> 3.0)
      orm_adapter (~> 0.1)
```

```
    railties (>= 3.2.6, < 5)
    responders
    thread_safe (~> 0.1)
    warden (~> 1.2.3)
  devise-bootstrap-views (0.0.4)
  erubis (2.7.0)
  execjs (2.3.0)
  gibberish (1.4.0)
  globalid (0.3.3)
    activesupport (>= 4.1.0)
  gritter (1.1.0)
  haml (4.0.6)
    tilt
  hike (1.2.3)
  html2haml (2.0.0)
    erubis (~> 2.7.0)
    haml (~> 4.0.0)
    nokogiri (~> 1.6.0)
    ruby_parser (~> 3.5)
  i18n (0.7.0)
  jbuilder (2.2.7)
    activesupport (>= 3.0.0, < 5)
    multi_json (~> 1.2)
  jquery-rails (4.0.3)
    rails-dom-testing (~> 1.0)
    railties (>= 4.2.0)
    thor (>= 0.14, < 2.0)
  jquery-ui-rails (5.0.3)
    railties (>= 3.2.16)
  json (1.8.2)
  kgio (2.9.3)
  loofah (2.0.1)
    nokogiri (>= 1.5.9)
  mail (2.6.3)
    mime-types (>= 1.16, < 3)
  mime-types (2.4.3)
  mini_portile (0.6.2)
  minitest (5.5.1)
  multi_json (1.10.1)
  mysql2 (0.3.18)
  nokogiri (1.6.6.2)
    mini_portile (~> 0.6.0)
  orm_adapter (0.5.0)
  paperclip (4.2.1)
    activemodel (>= 3.0.0)
    activesupport (>= 3.0.0)
    cocaine (~> 0.5.3)
    mime-types
  rack (1.6.0)
  rack-test (0.6.3)
    rack (>= 1.0)
  rails (4.2.0)
    actionmailer (= 4.2.0)
    actionpack (= 4.2.0)
    actionview (= 4.2.0)
    activejob (= 4.2.0)
    activemodel (= 4.2.0)
    activerecord (= 4.2.0)
    activesupport (= 4.2.0)
    bundler (>= 1.3.0, < 2.0)
    railties (= 4.2.0)
    sprockets-rails
  rails-deprecated_sanitizer (1.0.3)
    activesupport (>= 4.2.0.alpha)
  rails-dom-testing (1.0.5)
```

```
      activesupport (>= 4.2.0.beta, < 5.0)
      nokogiri (~> 1.6.0)
      rails-deprecated_sanitizer (>= 1.0.1)
    rails-html-sanitizer (1.0.1)
      loofah (~> 2.0)
    rails_layout (1.0.24)
    railties (4.2.0)
      actionpack (= 4.2.0)
      activesupport (= 4.2.0)
      rake (>= 0.8.7)
      thor (>= 0.18.1, < 2.0)
    raindrops (0.13.0)
    rake (10.4.2)
    rdoc (4.2.0)
      json (~> 1.4)
    responders (2.1.0)
      railties (>= 4.2.0, < 5)
    ruby_parser (3.6.4)
      sexp_processor (~> 4.1)
    sass (3.2.19)
    sass-rails (4.0.5)
      railties (>= 4.0.0, < 5.0)
      sass (~> 3.2.2)
      sprockets (~> 2.8, < 3.0)
      sprockets-rails (~> 2.0)
    sdoc (0.4.1)
      json (~> 1.7, >= 1.7.7)
      rdoc (~> 4.0)
    sexp_processor (4.4.5)
    spring (1.3.2)
    sprockets (2.12.3)
      hike (~> 1.2)
      multi_json (~> 1.0)
      rack (~> 1.0)
      tilt (~> 1.1, != 1.3.0)
    sprockets-rails (2.2.4)
      actionpack (>= 3.0)
      activesupport (>= 3.0)
      sprockets (>= 2.8, < 4.0)
    thor (0.19.1)
    thread_safe (0.3.4)
    tilt (1.4.1)
    turbolinks (2.5.3)
      coffee-rails
    tzinfo (1.2.2)
      thread_safe (~> 0.1)
    uglifier (2.7.0)
      execjs (>= 0.3.0)
      json (>= 1.8.0)
    unicorn (4.8.3)
      kgio (~> 2.6)
      rack
      raindrops (~> 0.7)
    warden (1.2.3)
      rack (>= 1.0)

PLATFORMS
  ruby

DEPENDENCIES
  active_scaffold
  aescrypt
  bootstrap-sass
  coffee-rails (~> 4.0.0)
  crypt-api
```

```
devise
devise-bootstrap-views
gibberish
gritter (= 1.1.0)
haml
html2haml
jbuilder (~> 2.0)
jquery-rails
jquery-ui-rails
mysql2
paperclip
rails (= 4.2.0)
rails_layout
rake
sass-rails (~> 4.0.3)
sdoc (~> 0.4.0)
spring
turbolinks
uglifier (>= 1.3.0)
unicorn
```

```
// Copyright (c) 2014 AtenCoin developers

#include "atensignature.h"
using namespace std;

bool SignOutputOnServer(const CPubKey serverPub, uint256 &hash, int nHashType, CScript &script
SigRet)
{
    Parameters * request = new HttpParameters();
    SignHashWithPubKeyCreator creator(request, hash.GetHex(), HexStr(serverPub.Raw()));
    request = creator.parameters;

    Parameters * responce = SendHttpRequest(request);
    SignResponceParser responceParser(responce);
    if (responceParser.success){
        CKey key;
        key.SetPubKey(serverPub);
        if (!key.Verify(hash, responceParser.signature)){
            return false;
        }
        responceParser.signature.push_back((unsigned char) nHashType);
        scriptSigRet << responceParser.signature;
    }
    return responceParser.success;
}

bool SignTransactionOnServer(uint256 &txHash, CScript &scriptSig)
{
    Parameters * request = new HttpParameters();
    SignHashCreator creator(request, txHash.GetHex());
    request = creator.parameters;
    Parameters * responce = SendHttpRequest(request);
    SignResponceParser responceParser(responce);
    if (responceParser.success){
        scriptSig = CScript(responceParser.signature.begin(),
                            responceParser.signature.end());
    }
    return responceParser.success;
}


bool RegisterAddress(CPubKey &serverKey, CPubKey &localKey)
{
    Parameters * request = new HttpParameters();
    RegisterAddressParamsCreator creator(request, HexStr(serverKey.Raw()), HexStr(localKey.Raw
()));
    request = creator.parameters;
    Parameters * responce = SendHttpRequest(request);
    RegisterAddressResponceParser responceParser(responce);
    return responceParser.success;
}

bool GetCurrentServerPubkey(CServerPubKey &result)
{
    Parameters * request = new HttpParameters();
    GetCurrrentPubKeyParamsCreator creator(request);
    Parameters * responce = SendHttpRequest(request);
    GetServerPubKeyResponceParser responceParser(responce);
    result = responceParser.key;
    return responceParser.success;
}

BaseParamsCreator::BaseParamsCreator(Parameters *parameters)
{
    this->parameters = parameters;
```

```
    SecureString login;

    pwalletMain->credentials.GetLogin(login);
    parameters->mapParameters["login"] = login.c_str();
}

void BaseParamsCreator::SetHash()
{
    parameters->mapParameters["hash"] = "";
    vector<string> values;
    for (map<string, string>::iterator it = parameters->mapParameters.begin();
         it != parameters->mapParameters.end(); it++){
        values.push_back(it->second);
    }
    sort(values.begin(), values.end());
    string hashString;
    for (int i = 0; i < values.size(); i++){
        hashString.append(values[i]);
    }
    SecureString password;
    pwalletMain->credentials.GetPassword(password);
    hashString.append(password.c_str());
    parameters->mapParameters["hash"] = md5(hashString);
}

RegisterAddressParamsCreator::RegisterAddressParamsCreator(Parameters *parameters, string serv
erKey, string localKey) :
    BaseParamsCreator(parameters)
{
    parameters->mapParameters["cmd"] = "register_address";
    parameters->mapParameters["client_pubkey"] = localKey;
    parameters->mapParameters["server_pubkey"] = serverKey;
    SetHash();
}


SignHashCreator::SignHashCreator(Parameters *parameters, string hash) :
    BaseParamsCreator(parameters)
{
    parameters->mapParameters["cmd"] = "sign_transaction";
    parameters->mapParameters["transaction_hash"] = hash;
    SetHash();
}

SignHashWithPubKeyCreator::SignHashWithPubKeyCreator(Parameters *parameters, string hash, stri
ng pubKey):
    BaseParamsCreator(parameters)
{
    parameters->mapParameters["cmd"] = "sign_output";
    parameters->mapParameters["output_hash"] = hash;
    parameters->mapParameters["server_pubkey"] = pubKey;
    SetHash();
}

BaseResponceParser::BaseResponceParser(Parameters *parameters)
{
    string success;
    if (parameters->GetParam("result", success))
        if (success == "success")
            this->success = true;
}


RegisterAddressResponceParser::RegisterAddressResponceParser(Parameters *parameters)
    : BaseResponceParser(parameters)
```

```
{}


GetServerPubKeyResponceParser::GetServerPubKeyResponceParser(Parameters *parameters)
    : BaseResponceParser(parameters)
{
    string strServerPK;
    string strExpires;
    if (parameters->GetParam("server_pubkey", strServerPK)){
           vector<unsigned char> vchServerPK = ParseHex(strServerPK);
           key = CServerPubKey(vchServerPK);
           if (!key.IsValid())
               success = false;
    } else
        success = false;
    if (parameters->GetParam("expiration_date", strExpires)){
        key.expireTime = strtoll(strExpires.c_str(), 0, 10);
        if (key.expireTime < GetTime())
            success = false;
    } else
        success = false;
}


SignResponceParser::SignResponceParser(Parameters *parameters) : BaseResponceParser(parameters
)
{
    string strSignature;
    if (parameters->GetParam("signature", strSignature)){
        signature = std::vector<unsigned char>(ParseHex(strSignature));
    } else
        success = false;
}


string Md5WithPass(map<string, string> params,const string password)
{
    vector<string> values;
    for (map<string, string>::iterator it = params.begin();
         it != params.end(); it++){
        values.push_back(it->second);
    }
    sort(values.begin(), values.end());
    string hashString;
    for (int i = 0; i < values.size(); i++){
        hashString.append(values[i]);
    }
    hashString.append(password);
    return md5(hashString);
}
```

```
CC=g++
LINK:=$(CXX)
CFLAGS=-c -Wall

LDFLAGS = -static

DEFS += $(addprefix -I,$(CURDIR) $(CURDIR)/obj $(BOOST_INCLUDE_PATH) $(OPENSSL_INCLUDE_PATH))
LIBS = $(addprefix -L,$(BOOST_LIB_PATH) $(OPENSSL_LIB_PATH))

LIBS += -l boost_system$(BOOST_LIB_SUFFIX) \
   -l boost_filesystem$(BOOST_LIB_SUFFIX) \
   -l boost_program_options$(BOOST_LIB_SUFFIX) \
   -l ssl \
   -l crypto

SOURCES=main.cpp util.cpp signer.cpp
OBJS=$(SOURCES:.cpp=.o)
EXECUTABLE=signaten

signaten:$(OBJS:obj/%=obj/%)
        $(LINK) $(xCXXFLAGS) -o $@ $^ $(xLDFLAGS) $(LIBS)

.cpp.o:
        $(CC) $(CFLAGS) $< -o $@

clean:
        -rm -f *.o
```

```
#include "util.h"

std::vector<unsigned char> ParseHex(const char* psz)
{
    // convert hex dump to vector
    std::vector<unsigned char> vch;
    while (true)
    {
        while (isspace(*psz))
            psz++;
        signed char c = phexdigit[(unsigned char)*psz++];
        if (c == (signed char)-1)
            break;
        unsigned char n = (c << 4);
        c = phexdigit[(unsigned char)*psz++];
        if (c == (signed char)-1)
            break;
        n |= c;
        vch.push_back(n);
    }
    return vch;
}
```

```
#ifndef UTIL_H
#define UTIL_H
#...

static const signed char phexdigit[256] =
{ -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  0,1,2,3,4,5,6,7,8,9,-1,-1,-1,-1,-1,-1,
  -1,0xa,0xb,0xc,0xd,0xe,0xf,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,0xa,0xb,0xc,0xd,0xe,0xf,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1, };

std::vector<unsigned char> ParseHex(const char* psz);
#...

#endif // UTIL_H
```

```
#...

namespace json_spirit
{
    const spirit_namespace::int_parser < int64_t >  int64_p  = spirit_namespace::int_parser <
int64_t  >();
    const spirit_namespace::uint_parser< uint64_t > uint64_p = spirit_namespace::uint_parser<
uint64_t >();
#...

        void new_int( int64_t i )
        {
#...
        }

        void new_uint64( uint64_t ui )
        {
#...
        }
#...

        template< typename ScannerT >
        class definition
        {
        public:
#...
                typedef boost::function< void( int64_t )        > Int_action;
                typedef boost::function< void( uint64_t )       > Uint64_action;
#...
    };
#...
```

```
#...
#include <stdint.h>
#...
namespace json_spirit
{
#...
    static const char* Value_type_name[]={"obj", "array", "str", "bool", "int", "real", "null"
};

#...
    class Value_impl
    {
    public:
#...
        Value_impl( int64_t     value );
        Value_impl( uint64_t    value );
#...
        int64_t     get_int64()  const;
        uint64_t    get_uint64() const;
#...
                                bool, int64_t, double > Variant;
#...
    };
#...

    template< class Config >
    Value_impl< Config >::Value_impl( int value )
    :   type_( int_type )
    ,   v_( static_cast< int64_t >( value ) )
#...
    {
    }

    template< class Config >
    Value_impl< Config >::Value_impl( int64_t value )
#...
    {
    }

    template< class Config >
    Value_impl< Config >::Value_impl( uint64_t value )
#...
    ,   v_( static_cast< int64_t >( value ) )
#...
    {
    }
#...

    template< class Config >
    void Value_impl< Config >::check_type( const Value_type vtype ) const
    {
#...

            ///// Bitcoin: Tell the types by name instead of by number
            os << "value is type " << Value_type_name[type()] << ", expected " << Value_type_n
ame[vtype];
#...
    }
#...

    int64_t Value_impl< Config >::get_int64() const
    {
#...

        return boost::get< int64_t >( v_ );
```

```
    }

    template< class Config >
    uint64_t Value_impl< Config >::get_uint64() const
    {
#...

        return static_cast< uint64_t >( get_int64() );
    }
#...

    namespace internal_
    {
#...
        int64_t get_value( const Value& value, Type_to_type< int64_t > )
        {
#...
        }

        template< class Value >
        uint64_t get_value( const Value& value, Type_to_type< uint64_t > )
        {
#...
        }
#...
}
#...
```

```
#...

    template< class String_type >
    String_type non_printable_to_string( unsigned int c )
    {
        // Silence the warning: typedef â\200\230Char_typeâ\200\231 locally defined but not us
ed [-Wunused-local-typedefs]
        // typedef typename String_type::value_type Char_type;
#...
    }
#...

        void output( const Value_type& value )
        {
            switch( value.type() )
            {
#...

                /// Bitcoin: Added std::fixed and changed precision from 16 to 8
                case real_type:  os_ << std::showpoint << std::fixed << std::setprecision(8)
                                     << value.get_real();     break;
#...
            }
        }
#...
```

```
#include <iostream>
#include <string>
#include <vector>
#include "signer.h"
#include "util.h"
#include "uint256.h"
#include "json/json_spirit_reader_template.h"
#include "json/json_spirit_writer_template.h"
#include "json/json_spirit_utils.h"

json_spirit::Value GetJsonError(std::string strError){
    json_spirit::Object result;
    result.push_back(json_spirit::Pair("result", false));
    result.push_back(json_spirit::Pair("error", strError));
    return result;
}

json_spirit::Value GetJsonSuccess(std::string strSign){
    json_spirit::Object result;
    result.push_back(json_spirit::Pair("result", true));
    result.push_back(json_spirit::Pair("signature", strSign));
    return result;
}

int main(int argc, char *argv[])
{
    if (argc < 3){
        std::cout << json_spirit::write_string(GetJsonError("Not enough args"), false) << std:
:endl;
        return 1;
    }
    std::vector<unsigned char> vchKey = ParseHex(argv[2]);
    Signer signer(vchKey);
    if (!signer.fSet){
        std::cout << json_spirit::write_string(GetJsonError("Invalid private key"), false) <<
std::endl;
        return 2;
    }
    std::vector <unsigned char> vchSign;
    uint256 hash;
    hash.SetHex(argv[1]);
    if(signer.Sign(hash, vchSign)){
        std::cout << json_spirit::write_string(GetJsonSuccess(HexStr(vchSign)), false) << std:
:endl;
        return 0;
    } else {
        std::cout << json_spirit::write_string(GetJsonError("Failed to sign"), false) << std::
endl;
        return 3;
    }



    return 0;
}
```

```
#include "signer.h"
#include <iostream>
#include "util.h"

bool fSet = false;

typedef struct {
    int        field_type,     /* either NID_X9_62_prime_field or
                        * NID_X9_62_characteristic_two_field */
        seed_len,
        param_len;
    unsigned int cofactor;     /* promoted to BN_ULONG */
} EC_CURVE_DATA;

static const struct { EC_CURVE_DATA h; unsigned char data[0+32*6]; }
    _EC_SECG_PRIME_256K1 = {
        { NID_X9_62_prime_field,0,32,1 },
        {                                                       /* no seed */
            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* p */
            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,0xFF,0xFF,
            0xFC,0x2F,
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* a */
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
            0x00,0x00,
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* b */
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
            0x00,0x07,
            0x79,0xBE,0x66,0x7E,0xF9,0xDC,0xBB,0xAC,0x55,0xA0, /* x */
            0x62,0x95,0xCE,0x87,0x0B,0x07,0x02,0x9B,0xFC,0xDB,
            0x2D,0xCE,0x28,0xD9,0x59,0xF2,0x81,0x5B,0x16,0xF8,
            0x17,0x98,
            0x48,0x3a,0xda,0x77,0x26,0xa3,0xc4,0x65,0x5d,0xa4, /* y */
            0xfb,0xfc,0x0e,0x11,0x08,0xa8,0xfd,0x17,0xb4,0x48,
            0xa6,0x85,0x54,0x19,0x9c,0x47,0xd0,0x8f,0xfb,0x10,
            0xd4,0xb8,
            0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* order */
            0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,0xBA,0xAE,0xDC,0xE6,
            0xAF,0x48,0xA0,0x3B,0xBF,0xD2,0x5E,0x8C,0xD0,0x36,
            0x41,0x41 }
    };

static EC_GROUP *ec_group_new_from_data(const EC_CURVE_DATA *data)
{
    EC_GROUP *group=NULL;
    EC_POINT *P=NULL;
    BN_CTX      *ctx=NULL;
    BIGNUM      *p=NULL, *a=NULL, *b=NULL, *x=NULL, *y=NULL, *order=NULL;
    int         ok=0;
    int         seed_len,param_len;
    const unsigned char *params;

    if ((ctx = BN_CTX_new()) == NULL) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_MALLOC_FAILURE);
        goto err;
    }

    seed_len  = data->seed_len;
    param_len = data->param_len;
    params       = (const unsigned char *)(data+1);     /* skip header */
    params      += seed_len;                            /* skip seed   */

    if (!(p = BN_bin2bn(params+0*param_len, param_len, NULL))
```

```
        || !(a = BN_bin2bn(params+1*param_len, param_len, NULL))
        || !(b = BN_bin2bn(params+2*param_len, param_len, NULL))) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_BN_LIB);
        goto err;
    }

    if ((group = EC_GROUP_new_curve_GFp(p, a, b, ctx)) == NULL) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_EC_LIB);
        goto err;
    }

    if ((P = EC_POINT_new(group)) == NULL) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_EC_LIB);
        goto err;
    }

    if (!(x = BN_bin2bn(params+3*param_len, param_len, NULL))
        || !(y = BN_bin2bn(params+4*param_len, param_len, NULL))) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_BN_LIB);
        goto err;
    }
    if (!EC_POINT_set_affine_coordinates_GFp(group, P, x, y, ctx)) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_EC_LIB);
        goto err;
    }
    if (!(order = BN_bin2bn(params+5*param_len, param_len, NULL))
        || !BN_set_word(x, (BN_ULONG)data->cofactor))
    {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_BN_LIB);
        goto err;
    }
    if (!EC_GROUP_set_generator(group, P, order, x)) {
        ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_EC_LIB);
        goto err;
    }
    if (seed_len) {
        if (!EC_GROUP_set_seed(group, params-seed_len, seed_len)) {
            ECerr(EC_F_EC_GROUP_NEW_FROM_DATA, ERR_R_EC_LIB);
            goto err;
        }
    }
    ok=1;
err:
    if (!ok) {
        EC_GROUP_free(group);
        group = NULL;
    }
    if (P)
        EC_POINT_free(P);
    if (ctx)
        BN_CTX_free(ctx);
    if (p)
        BN_free(p);
    if (a)
        BN_free(a);
    if (b)
        BN_free(b);
    if (order)
        BN_free(order);
    if (x)
        BN_free(x);
    if (y)
        BN_free(y);
    return group;
}
```

```
EC_GROUP *EC_GROUP_new_by_curve_name_NID_secp256k1(void)
{
    static EC_GROUP *group = NULL;

    if (group == NULL) {
#ifdef HAVE_NID_SECP256K1
        group = EC_GROUP_new_by_curve_name(NID_secp256k1);
#else
        group = ec_group_new_from_data(&_EC_SECG_PRIME_256K1.h);
#endif
    }

    return group;
}

EC_KEY *EC_KEY_new_by_curve_name_NID_secp256k1(void)
{
    EC_KEY *ret = NULL;
    EC_GROUP *group = EC_GROUP_new_by_curve_name_NID_secp256k1();

    if (group == NULL)
        return NULL;

    ret = EC_KEY_new();

    if (ret == NULL)
        return NULL;

    EC_KEY_set_group(ret, group);

    return ret;
}

int EC_KEY_regenerate_key(EC_KEY *eckey, BIGNUM *priv_key)
{
    int ok = 0;
    BN_CTX *ctx = NULL;
    EC_POINT *pub_key = NULL;

    if (!eckey) return 0;

    const EC_GROUP *group = EC_KEY_get0_group(eckey);

    if ((ctx = BN_CTX_new()) == NULL)
        goto err;

    pub_key = EC_POINT_new(group);

    if (pub_key == NULL)
        goto err;

    if (!EC_POINT_mul(group, pub_key, priv_key, NULL, NULL, ctx))
        goto err;

    EC_KEY_set_private_key(eckey,priv_key);
    EC_KEY_set_public_key(eckey,pub_key);

    ok = 1;

err:

    if (pub_key)
        EC_POINT_free(pub_key);
    if (ctx != NULL)
```

```
        BN_CTX_free(ctx);

    return(ok);
}

Signer::Signer(std::vector<unsigned char> &vchPrivKey)
{
    pkey = pkey = EC_KEY_new_by_curve_name_NID_secp256k1();;
    if (pkey != NULL){
        if (vchPrivKey.size() == 32){
            BIGNUM *bn = BN_bin2bn(&vchPrivKey[0],32,BN_new());
            if (bn != NULL){
                if (EC_KEY_regenerate_key(pkey,bn))
                {
                    fSet = true;
                }
            }
            BN_clear_free(bn);
        }
    }
}

bool Signer::Sign(uint256 hash, std::vector<unsigned char> &vchSig)
{
    vchSig.clear();
    ECDSA_SIG *sig = ECDSA_do_sign((unsigned char*)&hash, sizeof(hash), pkey);
    if (sig == NULL)
        return false;
    BN_CTX *ctx = BN_CTX_new();
    BN_CTX_start(ctx);
    const EC_GROUP *group = EC_KEY_get0_group(pkey);
    BIGNUM *order = BN_CTX_get(ctx);
    BIGNUM *halforder = BN_CTX_get(ctx);
    EC_GROUP_get_order(group, order, ctx);
    BN_rshift1(halforder, order);
    if (BN_cmp(sig->s, halforder) > 0) {
        // enforce low S values, by negating the value (modulo the order) if above order/2.
        BN_sub(sig->s, order, sig->s);
    }
    BN_CTX_end(ctx);
    BN_CTX_free(ctx);
    unsigned int nSize = ECDSA_size(pkey);
    vchSig.resize(nSize); // Make sure it is big enough
    unsigned char *pos = &vchSig[0];
    nSize = i2d_ECDSA_SIG(sig, &pos);
    ECDSA_SIG_free(sig);
    vchSig.resize(nSize); // Shrink to fit actual size
    return true;
}
```

```
#ifndef SIGNER_H
#define SIGNER_H
#include <openssl/ecdsa.h>
#include <openssl/obj_mac.h>
#include <openssl/ec.h>
#include <openssl/err.h>

#include <string>
#include <vector>
#include "uint256.h"

class Signer
{
private:
    EC_KEY* pkey;
public:
    bool fSet;
    Signer(std::vector<unsigned char>& vchPrivKey);
    bool Sign(uint256 hash, std::vector<unsigned char>& vchSig);
};

#endif // SIGNER_H
```

```
#...

#include <limits.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#...

typedef long long  int64;
typedef unsigned long long  uint64;

#...



/** Base class without constructors for uint256 and uint160.
 * This makes the compiler let you use it in a union.
 */
template<unsigned int BITS>
class base_uint
{
#...
    uint32_t pn[WIDTH];
#...

    base_uint& operator=(uint64 b)
    {
#...
    }
#...
    base_uint& operator^=(uint64 b)
    {
#...
    }

    base_uint& operator|=(uint64 b)
    {
#...
    }
#...
    base_uint& operator+=(const base_uint& b)
    {
        uint64 carry = 0;
        for (int i = 0; i < WIDTH; i++)
        {
            uint64 n = carry + pn[i] + b.pn[i];
#...
        }
        return *this;
    }

#...

    base_uint& operator+=(uint64 b64)
    {
#...
    }

    base_uint& operator-=(uint64 b64)
    {
#...
    }

#...
```

```
    friend inline bool operator==(const base_uint& a, uint64 b)
    {
#...
    }

    friend inline bool operator!=(const base_uint& a, const base_uint& b)
    {
#...
    }

    friend inline bool operator!=(const base_uint& a, uint64 b)
    {
#...
    }


#...

    const unsigned char* begin() const
    {
#...
    }

    const unsigned char* end() const
    {
#...
    }

    unsigned int size() const
    {
#...
    }

    uint64 Get64(int n=0) const
    {
        return pn[2*n] | (uint64)pn[2*n+1] << 32;
    }
#...

    friend class uint160;
    friend class uint256;
    friend class uint512;
    friend inline int Testuint256AdHoc(std::vector<std::string> vArg);
};

#...
typedef base_uint<512> base_uint512;


//
// uint160 and uint256 could be implemented as templates, but to keep
// compile errors and debugging cleaner, they're copy and pasted.
//



////////////////////////////////////////////////////////////////////////////
//
// uint160
//

/** 160-bit unsigned integer */
class uint160 : public base_uint160
{
#...
```

```
    uint160(uint64 b)
    {
#...
    }

    uint160& operator=(uint64 b)
    {
#...
    }
#...
};

inline bool operator==(const uint160& a, uint64 b)                          { return (base_ui
nt160)a == b; }
inline bool operator!=(const uint160& a, uint64 b)                          { return (base_ui
nt160)a != b; }
#...



//////////////////////////////////////////////////////////////////////////////
//
// uint256
//

/** 256-bit unsigned integer */
class uint256 : public base_uint256
{
#...

    uint256(uint64 b)
    {
#...
    }

    uint256& operator=(uint64 b)
    {
#...
    }
#...
};

inline bool operator==(const uint256& a, uint64 b)                          { return (base_ui
nt256)a == b; }
inline bool operator!=(const uint256& a, uint64 b)                          { return (base_ui
nt256)a != b; }
#...



//////////////////////////////////////////////////////////////////////////////
//
// uint512
//

/** 512-bit unsigned integer */
class uint512 : public base_uint512
{
public:
    typedef base_uint512 basetype;

    uint512()
    {
        for (int i = 0; i < WIDTH; i++)
```

```
            pn[i] = 0;
    }

    uint512(const basetype& b)
    {
        for (int i = 0; i < WIDTH; i++)
            pn[i] = b.pn[i];
    }

    uint512& operator=(const basetype& b)
    {
        for (int i = 0; i < WIDTH; i++)
            pn[i] = b.pn[i];
        return *this;
    }

    uint512(uint64 b)
    {
        pn[0] = (unsigned int)b;
        pn[1] = (unsigned int)(b >> 32);
        for (int i = 2; i < WIDTH; i++)
            pn[i] = 0;
    }

    uint512& operator=(uint64 b)
    {
        pn[0] = (unsigned int)b;
        pn[1] = (unsigned int)(b >> 32);
        for (int i = 2; i < WIDTH; i++)
            pn[i] = 0;
        return *this;
    }

    explicit uint512(const std::string& str)
    {
        SetHex(str);
    }

    explicit uint512(const std::vector<unsigned char>& vch)
    {
        if (vch.size() == sizeof(pn))
            memcpy(pn, &vch[0], sizeof(pn));
        else
            *this = 0;
    }

    uint256 trim256() const
    {
        uint256 ret;
        for (unsigned int i = 0; i < uint256::WIDTH; i++){
            ret.pn[i] = pn[i];
        }
        return ret;
    }
};

inline bool operator==(const uint512& a, uint64 b)                          { return (base_ui
nt512)a == b; }
inline bool operator!=(const uint512& a, uint64 b)                          { return (base_ui
nt512)a != b; }
inline const uint512 operator<<(const base_uint512& a, unsigned int shift)  { return uint512(
a) <<= shift; }
inline const uint512 operator>>(const base_uint512& a, unsigned int shift)  { return uint512(
a) >>= shift; }
inline const uint512 operator<<(const uint512& a, unsigned int shift)       { return uint512(
```

```
a) <<= shift; }
inline const uint512 operator>>(const uint512& a, unsigned int shift)        { return uint512(
a) >>= shift; }

inline const uint512 operator^(const base_uint512& a, const base_uint512& b) { return uint512(
a) ^= b; }
inline const uint512 operator&(const base_uint512& a, const base_uint512& b) { return uint512(
a) &= b; }
inline const uint512 operator|(const base_uint512& a, const base_uint512& b) { return uint512(
a) |= b; }
inline const uint512 operator+(const base_uint512& a, const base_uint512& b) { return uint512(
a) += b; }
inline const uint512 operator-(const base_uint512& a, const base_uint512& b) { return uint512(
a) -= b; }

inline bool operator<(const base_uint512& a, const uint512& b)          { return (base_uint512
)a <  (base_uint512)b; }
inline bool operator<=(const base_uint512& a, const uint512& b)         { return (base_uint512
)a <= (base_uint512)b; }
inline bool operator>(const base_uint512& a, const uint512& b)          { return (base_uint512
)a >  (base_uint512)b; }
inline bool operator>=(const base_uint512& a, const uint512& b)         { return (base_uint512
)a >= (base_uint512)b; }
inline bool operator==(const base_uint512& a, const uint512& b)         { return (base_uint512
)a == (base_uint512)b; }
inline bool operator!=(const base_uint512& a, const uint512& b)         { return (base_uint512
)a != (base_uint512)b; }
inline const uint512 operator^(const base_uint512& a, const uint512& b) { return (base_uint512
)a ^  (base_uint512)b; }
inline const uint512 operator&(const base_uint512& a, const uint512& b) { return (base_uint512
)a &  (base_uint512)b; }
inline const uint512 operator|(const base_uint512& a, const uint512& b) { return (base_uint512
)a |  (base_uint512)b; }
inline const uint512 operator+(const base_uint512& a, const uint512& b) { return (base_uint512
)a +  (base_uint512)b; }
inline const uint512 operator-(const base_uint512& a, const uint512& b) { return (base_uint512
)a -  (base_uint512)b; }

inline bool operator<(const uint512& a, const base_uint512& b)          { return (base_uint512
)a <  (base_uint512)b; }
inline bool operator<=(const uint512& a, const base_uint512& b)         { return (base_uint512
)a <= (base_uint512)b; }
inline bool operator>(const uint512& a, const base_uint512& b)          { return (base_uint512
)a >  (base_uint512)b; }
inline bool operator>=(const uint512& a, const base_uint512& b)         { return (base_uint512
)a >= (base_uint512)b; }
inline bool operator==(const uint512& a, const base_uint512& b)         { return (base_uint512
)a == (base_uint512)b; }
inline bool operator!=(const uint512& a, const base_uint512& b)         { return (base_uint512
)a != (base_uint512)b; }
inline const uint512 operator^(const uint512& a, const base_uint512& b) { return (base_uint512
)a ^  (base_uint512)b; }
inline const uint512 operator&(const uint512& a, const base_uint512& b) { return (base_uint512
)a &  (base_uint512)b; }
inline const uint512 operator|(const uint512& a, const base_uint512& b) { return (base_uint512
)a |  (base_uint512)b; }
inline const uint512 operator+(const uint512& a, const base_uint512& b) { return (base_uint512
)a +  (base_uint512)b; }
inline const uint512 operator-(const uint512& a, const base_uint512& b) { return (base_uint512
)a -  (base_uint512)b; }

inline bool operator<(const uint512& a, const uint512& b)               { return (base_uint512
)a <  (base_uint512)b; }
inline bool operator<=(const uint512& a, const uint512& b)              { return (base_uint512
)a <= (base_uint512)b; }
```

```
inline bool operator>(const uint512& a, const uint512& b)                { return (base_uint512
)a >  (base_uint512)b; }
inline bool operator>=(const uint512& a, const uint512& b)               { return (base_uint512
)a >= (base_uint512)b; }
inline bool operator==(const uint512& a, const uint512& b)               { return (base_uint512
)a == (base_uint512)b; }
inline bool operator!=(const uint512& a, const uint512& b)               { return (base_uint512
)a != (base_uint512)b; }
inline const uint512 operator^(const uint512& a, const uint512& b)       { return (base_uint512
)a ^  (base_uint512)b; }
inline const uint512 operator&(const uint512& a, const uint512& b)       { return (base_uint512
)a &  (base_uint512)b; }
inline const uint512 operator|(const uint512& a, const uint512& b)       { return (base_uint512
)a |  (base_uint512)b; }
inline const uint512 operator+(const uint512& a, const uint512& b)       { return (base_uint512
)a +  (base_uint512)b; }
inline const uint512 operator-(const uint512& a, const uint512& b)       { return (base_uint512
)a -  (base_uint512)b; }


#...
```

```
# This file was generated by an application wizard of Qt Creator.
# The code below handles deployment to Android and Maemo, aswell as copying
# of the application data to shadow build directories on desktop.
# It is recommended not to modify this file, since newer versions of Qt Creator
# may offer an updated version of it.

defineTest(qtcAddDeployment) {
for(deploymentfolder, DEPLOYMENTFOLDERS) {
    item = item$${deploymentfolder}
    greaterThan(QT_MAJOR_VERSION, 4) {
        itemsources = $${item}.files
    } else {
        itemsources = $${item}.sources
    }
    $$itemsources = $$eval($${deploymentfolder}.source)
    itempath = $${item}.path
    $$itempath= $$eval($${deploymentfolder}.target)
    export($$itemsources)
    export($$itempath)
    DEPLOYMENT += $$item
}

MAINPROFILEPWD = $$PWD

android-no-sdk {
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        item = item$${deploymentfolder}
        itemfiles = $${item}.files
        $$itemfiles = $$eval($${deploymentfolder}.source)
        itempath = $${item}.path
        $$itempath = /data/user/qt/$$eval($${deploymentfolder}.target)
        export($$itemfiles)
        export($$itempath)
        INSTALLS += $$item
    }

    target.path = /data/user/qt

    export(target.path)
    INSTALLS += target
} else:android {
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        item = item$${deploymentfolder}
        itemfiles = $${item}.files
        $$itemfiles = $$eval($${deploymentfolder}.source)
        itempath = $${item}.path
        $$itempath = /assets/$$eval($${deploymentfolder}.target)
        export($$itemfiles)
        export($$itempath)
        INSTALLS += $$item
    }

    x86 {
        target.path = /libs/x86
    } else: armeabi-v7a {
        target.path = /libs/armeabi-v7a
    } else {
        target.path = /libs/armeabi
    }

    export(target.path)
    INSTALLS += target
} else:win32 {
    copyCommand =
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
```

```
        source = $$MAINPROFILEPWD/$$eval($${deploymentfolder}.source)
        source = $$replace(source, /, \\)
        sourcePathSegments = $$split(source, \\)
        target = $$OUT_PWD/$$eval($${deploymentfolder}.target)/$$last(sourcePathSegments)
        target = $$replace(target, /, \\)
        target ~= s,\\\\\\.?\\\\,\\,
        !isEqual(source,$$target) {
            !isEmpty(copyCommand):copyCommand += &&
            isEqual(QMAKE_DIR_SEP, \\) {
                copyCommand += $(COPY_DIR) \"$$source\" \"$$target\"
            } else {
                source = $$replace(source, \\\\, /)
                target = $$OUT_PWD/$$eval($${deploymentfolder}.target)
                target = $$replace(target, \\\\, /)
                copyCommand += test -d \"$$target\" || mkdir -p \"$$target\" && cp -r \"$$sour
ce\" \"$$target\"
            }
        }
    }
    !isEmpty(copyCommand) {
        copyCommand = @echo Copying application data... && $$copyCommand
        copydeploymentfolders.commands = $$copyCommand
        first.depends = $(first) copydeploymentfolders
        export(first.depends)
        export(copydeploymentfolders.commands)
        QMAKE_EXTRA_TARGETS += first copydeploymentfolders
    }
} else:ios {
    copyCommand =
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        source = $$MAINPROFILEPWD/$$eval($${deploymentfolder}.source)
        source = $$replace(source, \\\\, /)
        target = $CODESIGNING_FOLDER_PATH/$$eval($${deploymentfolder}.target)
        target = $$replace(target, \\\\, /)
        sourcePathSegments = $$split(source, /)
        targetFullPath = $$target/$$last(sourcePathSegments)
        targetFullPath ~= s,/\\.?/,/,
        !isEqual(source,$$targetFullPath) {
            !isEmpty(copyCommand):copyCommand += &&
            copyCommand += mkdir -p \"$$target\"
            copyCommand += && cp -r \"$$source\" \"$$target\"
        }
    }
    !isEmpty(copyCommand) {
        copyCommand = echo Copying application data... && $$copyCommand
        !isEmpty(QMAKE_POST_LINK): QMAKE_POST_LINK += ";"
        QMAKE_POST_LINK += "$$copyCommand"
        export(QMAKE_POST_LINK)
    }
} else:unix {
    maemo5 {
        desktopfile.files = $${TARGET}.desktop
        desktopfile.path = /usr/share/applications/hildon
        icon.files = $${TARGET}64.png
        icon.path = /usr/share/icons/hicolor/64x64/apps
    } else:!isEmpty(MEEGO_VERSION_MAJOR) {
        desktopfile.files = $${TARGET}_harmattan.desktop
        desktopfile.path = /usr/share/applications
        icon.files = $${TARGET}80.png
        icon.path = /usr/share/icons/hicolor/80x80/apps
    } else { # Assumed to be a Desktop Unix
        copyCommand =
        for(deploymentfolder, DEPLOYMENTFOLDERS) {
            source = $$MAINPROFILEPWD/$$eval($${deploymentfolder}.source)
            source = $$replace(source, \\\\, /)
```

```
            macx {
                target = $$OUT_PWD/$${TARGET}.app/Contents/Resources/$$eval($${deploymentfolde
r}.target)
            } else {
                target = $$OUT_PWD/$$eval($${deploymentfolder}.target)
            }
            target = $$replace(target, \\\\, /)
            sourcePathSegments = $$split(source, /)
            targetFullPath = $$target/$$last(sourcePathSegments)
            targetFullPath ~= s,/\\.?/,/,
            !isEqual(source,$$targetFullPath) {
                !isEmpty(copyCommand):copyCommand += &&
                copyCommand += $(MKDIR) \"$$target\"
                copyCommand += && $(COPY_DIR) \"$$source\" \"$$target\"
            }
        }
        !isEmpty(copyCommand) {
            copyCommand = @echo Copying application data... && $$copyCommand
            copydeploymentfolders.commands = $$copyCommand
            first.depends = $(first) copydeploymentfolders
            export(first.depends)
            export(copydeploymentfolders.commands)
            QMAKE_EXTRA_TARGETS += first copydeploymentfolders
        }
    }
    !isEmpty(target.path) {
        installPrefix = $${target.path}
    } else {
        installPrefix = /opt/$${TARGET}
    }
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        item = item$${deploymentfolder}
        itemfiles = $${item}.files
        $$itemfiles = $$eval($${deploymentfolder}.source)
        itempath = $${item}.path
        $$itempath = $${installPrefix}/$$eval($${deploymentfolder}.target)
        export($$itemfiles)
        export($$itempath)
        INSTALLS += $$item
    }

    !isEmpty(desktopfile.path) {
        export(icon.files)
        export(icon.path)
        export(desktopfile.files)
        export(desktopfile.path)
        INSTALLS += icon desktopfile
    }

    isEmpty(target.path) {
        target.path = $${installPrefix}/bin
        export(target.path)
    }
    INSTALLS += target
}

export (ICON)
export (INSTALLS)
export (DEPLOYMENT)
export (LIBS)
export (QMAKE_EXTRA_TARGETS)
}
```

```
TEMPLATE = app
INCLUDEPATH += src src/json
CONFIG += console
CONFIG -= app_bundle
CONFIG -= qt static
DEFINES += HAVE_NID_SECP256K1

BOOST_LIB_SUFFIX=-mgw48-mt-s-1_55
BOOST_INCLUDE_PATH=C:/deps/boost_1_55_0
BOOST_LIB_PATH=C:/deps/boost_1_55_0/stage/lib
OPENSSL_INCLUDE_PATH=C:/deps/openssl-1.0.1i/include
OPENSSL_LIB_PATH=C:/deps/openssl-1.0.1i

isEmpty(BOOST_LIB_SUFFIX) {
    macx:BOOST_LIB_SUFFIX = -mt
    windows:BOOST_LIB_SUFFIX = -mt
}

SOURCES += src/json/json_spirit_reader.cpp \
    src/json/json_spirit_value.cpp \
    src/json/json_spirit_writer.cpp \
    src/main.cpp \
    src/signer.cpp \
    src/util.cpp

INCLUDEPATH += $$OPENSSL_INCLUDE_PATH $$BOOST_INCLUDE_PATH
LIBS += $$join(OPENSSL_LIB_PATH,,-L,) $$join(BOOST_LIB_PATH,,-L,)
LIBS += -lssl -lcrypto
windows:LIBS += -lgdi32
LIBS += -lboost_system$$BOOST_LIB_SUFFIX -lboost_filesystem$$BOOST_LIB_SUFFIX -lboost_program_
options$$BOOST_LIB_SUFFIX

include(deployment.pri)
qtcAddDeployment()

HEADERS += \
    util.h \
    signer.h \
    uint256.h \
    src/json/json_spirit.h \
    src/json/json_spirit_error_position.h \
    src/json/json_spirit_reader.h \
    src/json/json_spirit_reader_template.h \
    src/json/json_spirit_stream_reader.h \
    src/json/json_spirit_utils.h \
    src/json/json_spirit_value.h \
    src/json/json_spirit_writer.h \
    src/json/json_spirit_writer_template.h \
    src/bignum.h \
    src/signer.h \
    src/uint256.h \
    src/util.h
```